EUROGRAPHICS 2014 / E. Galin and M. Wand *Short Paper*

Regional Time Stepping for SPH

Prashant Goswami[†1] Christopher Batty[‡2]

[1] INRIA, LJK, Université de Grenoble, France [§] [2] University of Waterloo, Canada

Abstract

This paper presents novel and efficient strategies to spatially adapt the amount of computational effort applied based on the local dynamics of a free surface flow, for classic weakly compressible SPH (WCSPH). Using a convenient and readily parallelizable block-based approach, different regions of the fluid are assigned differing time steps and solved at different rates to minimize computational cost. We demonstrate that our approach can achieve about two times speed-up over the standard method even in highly dynamic scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The *Smoothed Particle Hydrodynamics (SPH)* method is a powerful and widely used approach to liquid animation [MCG03, Mon05, BT07]; among other benefits, it produces detailed splashing and droplet effects, supports seamless topological changes and preservation of liquid mass, and handles complex boundaries in a straightforward manner. However, capturing a sufficiently wide range of spatial scales in order to generate visually compelling results often requires large particle counts, and correspondingly long simulation times.

Across all SPH methods, the choice of time step remains a crucial factor in determining the overall computational cost. Although recently developed methods target acceleration based on multi-resolution [APKG07, ZSP08, SG11], incompressibility [SP09, ICS*13], GPU-based acceleration [TH07, ZSP08, GSSP10] or adaptive global time-stepping [IAGT10, GP11], the possibility of using different time steps for different regions of fluid has not been explored.

In this paper, we introduce the *regional time stepping* (RTS) approach for WCSPH method, in which computational effort is expended on different fluid regions in proportion to the speed of their local dynamics. In our numerical experiments, we were able to reduce simulation times by approximately a factor of two compared to the standard method on realistic, highly dynamic scenes in which the entire connected body of fluid is in motion. Our algorithm relies on an efficient block-based technique to determine the different regions and support convenient parallelism. Further, our method could be easily integrated with existing techniques such as GPU-methods or multi-scale simulation to provide added benefit.

2. Block-based Computation

Our algorithm relies on a block-based architecture. If s is the initial particle spacing, we divide the simulation domain into a virtual grid, with each block having support radius r, such that $r \simeq 2s$. Thus each particle is contained by exactly one of the blocks in the simulation domain.

Such an arrangement has several benefits. For example, neighbors of all particles in a block can be computed efficiently by examining neighboring blocks. Each block can also be treated as a parallelization unit for computing the physics of particles within it, as in the work of Goswami et al. [GSSP10].

However, the most important advantage of the block-based arrangement in our case is parallel region determination. The time steps for a given region are computed over these virtual blocks instead of at the particle level, under the reasonable assumption that liquid in a local area tends to be deforming at comparable rates. This method can then be efficiently parallelized by launching a thread per filled block

† prashant.goswami@inria.fr
‡ christopher.batty@uwaterloo.ca
§ *previously NTU, Singapore*

instead of per particle. Particles falling within that block report their velocity and force up to the parent block, thereby avoiding any race or collision conditions.

Our simple block-based time step computation is illustrated in Figure 1, and comprises three steps:

1. All particles compute their velocity and total force.
2. Particles propagate their attributes to their parent (i.e., containing) block. A minimum time step is computed for the block based on the maximum force and velocity from its particles.
3. Each block's time step is propagated back to its particles.

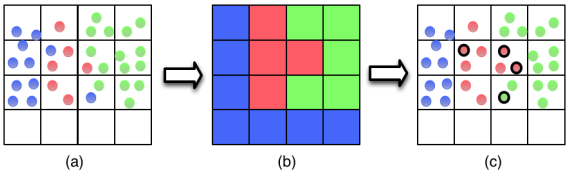

Figure 1: *Block-based time step computation for particles, red (Δt_b), green ($2\Delta t_b$) and blue ($3\Delta t_b$) where Δt_b is the base (smallest) time step. (a) Particles colored by the individual time steps they would ordinarily possess. (b) Particles pass their velocity and force values to their parent blocks. Each block is assigned the minimum required time step based on its particles. (c) The block propagates its computed time step back to the particles. (Particles whose time step has been altered are outlined in black.)*

In what follows, $\Re_n$ denotes a *region* or set of blocks assigned to a given time step $\Delta t_n = n\Delta t_b$ where Δt_b is the base time step, and n is a positive integer. The corresponding particle set is denoted by S_n.

A block is assigned to a region corresponding to the largest time step for which it satisfies a set of three criteria, according to its particles' maximum velocity and force. The first two criteria are:

$$\Delta t_n \leq \frac{\lambda_v r}{c_s} \tag{1}$$

$$\Delta t_n \leq \lambda_f \sqrt{\frac{rm}{F_{max}}} \tag{2}$$

These are standard time step conditions from the SPH literature (e.g., [DC99, BT07]); the first is a CFL condition, while the second accounts for sudden accelerations over a time step. In these equations, c_s is the speed of sound in the medium, m is the particle mass, F_{max} is the maximum force magnitude of particles in the block, and V_{max} is the maximum velocity magnitude of particles in the block. We set the remaining coefficients λ to $\lambda_v \leq 0.4$ and $\lambda_f \leq 0.25$.

We introduce a third criterion to partition particles into groups depending on their velocities:

$$\frac{\Delta t_n V_{max}}{r} \leq \alpha\beta_n \tag{3}$$

where,

$$\beta_1 = \infty, \quad \beta_n = 0.4(0.2)^{(n-2)} \text{ for } n \geq 2 \quad \alpha = 0.4$$

In essence, Equation 3 assigns to each particle a time step based on the fraction of its support radius that it would cover in a step moving at its current velocity.

3. Regional Time Stepping with SPH

Serna et al. [SRS03] introduced an asynchronous predictor-corrector time integration strategy for their DEVA astrophysical SPH code. Given a set of particles assigned different time steps, consider advancing through the union of all the resulting time steps. Beginning from a current time t_n, with positions x_i^n, velocities v_i^n, and accelerations a_i^n for each particle i, the following predictor step of length $\Delta t = t_{n+1} - t_n$ is taken by *all* particles to estimate new velocities and positions at time t_{n+1}:

$$\tilde{x}_i^{n+1} = x_i^n + v_i^n \Delta t + \frac{a_i^n (\Delta t)^2}{2} \tag{4}$$

$$\tilde{v}_i^{n+1} = v_i^n + a_i^n \Delta t \tag{5}$$

Among the set of all particles, the time t_{n+1} will be the conclusion of a "true" time step for some, called *active particles*; for the remainder this step is taken only to provide intermediate information to nearby particles. Next, only the active particles have their neighborhoods and accelerations re-evaluated at t_{n+1}, and their positions and velocities are corrected:

$$x_i^{n+1} = \tilde{x}_i^{n+1} + \frac{(a_i^{n+1} - a_i^n)\delta t^2}{6} \tag{6}$$

$$v_i^{n+1} = \tilde{v}_i^{n+1} + \frac{(a_i^{n+1} - a_i^n)\delta t}{2} \tag{7}$$

Although this approach saves on expensive evaluations of forces and accelerations, it still requires substepping of *all* particles in the simulation at the smallest global time step. We make the further observation that if all the particles within a given particle's neighborhood require only the same or larger time step, then no interpolated substeps need to be taken and the final result will be the same. Therefore in regions of our domain assigned large time steps, we can safely integrate all the contained particles at that timestep without the need to perform any substepping whatsoever. This allows the simulation to remain synchronized overall, while correctly integrating different regions at appropriate rates and avoiding unnecessary computation, see also Figure 2.

The basic outline of our approach is presented in Algorithm 1. The first step assigns a time step to each block

within the simulation domain. That is, we choose $\Re_n$ and S_n and update the global block-based neighborhood grid.

To ensure that the time step varies gradually across the physical domain, which aids in simulating quite stiff incompressible flows, we locate the boundary between regions with different time steps, and determine the set of blocks $\Re_{min}$ on the side with the larger time step. This region is then assigned the smaller time step of its neighboring region, which is done efficiently at the block level by checking each block's neighbors.

In our algorithm, the particles maintain a few additional variables. *validity* is the number of the smallest time steps for which its most recently computed attributes are assumed valid (i.e., how many substeps before its true time step ends). *compute* is a Boolean flag that indicates whether the particle is currently active (i.e., requires re-computation of its acceleration, and end-of-step correction of its position and velocity) which occurs when the $validity \leq 0$. If the particle is active, its neighborhood set is determined and its local density and forces are computed. Otherwise, it skips these steps. At the end of each loop, the position and velocity of each particle is updated, and active particles have their velocities and positions corrected (lines 25-30), per Serna's scheme [SRS03].

We make some additional observations. First, while the computation of time steps is determined per block, it is updated on the individual particles which also track their own validity. Blocks do not have validity or history, and therefore all computations over blocks are valid only for a frame. Second, the algorithm is pre-emptive. That is, a particle can change its time step even before its validity expires (line 14 of Algorithm 1). This allows the method to maintain stability in the face of sudden accelerations, as often occurs in collisions with boundaries. Figure 3 illustrates a double dam break simulated using this scheme.

4. Results

The proposed and baseline WCSPH were implemented on a machine with a 3.2 GHz quad-core Intel processor, using C++ and the OpenMP API. The images were rendered offline with POVRAY.

We have used Δt_{cfl} for standard time stepping and $\Delta t_b = \Delta t_{cfl}$ for RTS. Table 1 gives the performance speed-up of our method in comparison to WCSPH for two highly dynamic examples. Our method yields simulations about twice as fast as the comparison method for the given examples, even in situations where globally adaptive time-stepping cannot achieve benefit, for example in demo set-up for Figure 4.

5. Future Work

We presented an efficient block-based method for regional time stepping using WCSPH. An interesting direction would be to extend RTS for PCISPH / IISPH.

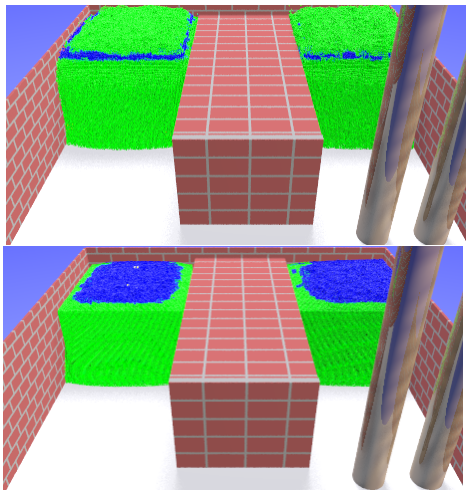

Figure 2: *The initial moments of the double dam break example using RTS WCSPH. Top: Without applying the second order velocity and position corrections of Serna et al., the particles quickly push apart and the smooth surface is disrupted. Bottom: With the corrections applied we achieve the expected behavior.*

	Scene	# particles	Speed up (Ours vs. Standard)
SPH	Figure 3	1.5M	2
	Figure 4	2.2M	1.8

Table 1: *Performance comparisons of regional time stepping with standard SPH.*

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (July 2007). 1

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), SCA '07, pp. 209–217. 1, 2

[DC99] DESBRUN M., CANI M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. Rep. 3829, INRIA, BP 105 - 78153 Le Chesnay Cedex - France, December 1999. 2

[GP11] GOSWAMI P., PAJAROLA R.: Time adaptive approximate SPH. In *Proceedings of the VRIPHYS* (2011), pp. 19–28. 1

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), SCA '10, pp. 55–64. 1

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *VRIPHYS* (2010), Erleben K., Bender J., Teschner M., (Eds.), Eurographics Association, pp. 79–88. 1

[ICS*13] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH.

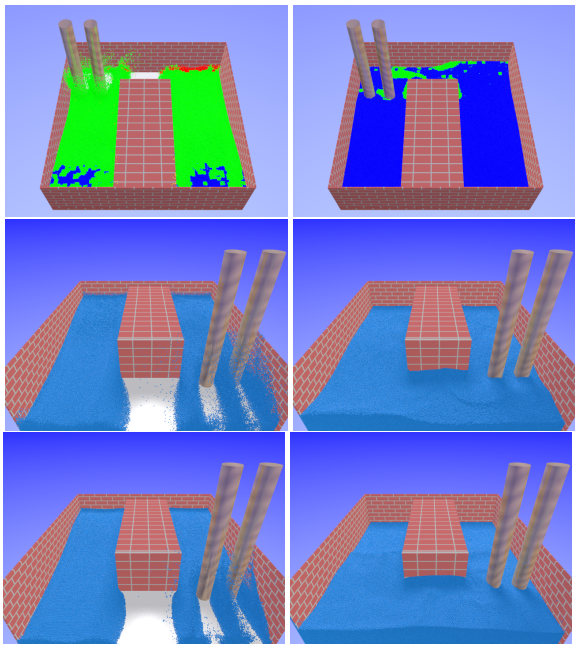

Figure 3: *A 1.5M particle double dam break simulated using our regional time stepping algorithm for WCSPH. Top: Particles colored by time step region. Middle: Particles for corresponding frames from a different viewpoint and colored uniformly. Bottom: Corresponding frames simulated using standard SPH simulation.*

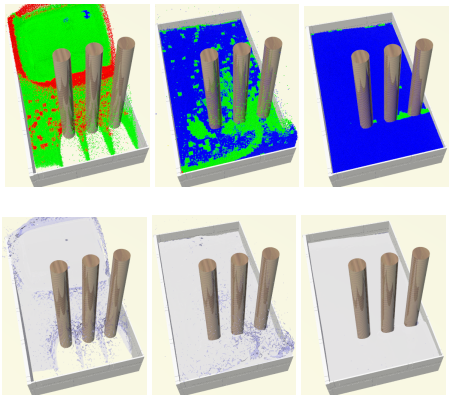

Figure 4: *A dam break scenario where immediately after releasing the dam we drop a second block of liquid on top, simulated with 2.2M particles using RTS WCSPH.*

Algorithm 1 Regional Time Stepping for Standard SPH

for all particles i **do**
  set $validity_i = 0$

while (animating) **do**
  update block neighborhood grid

  /*——— Region determination($\Re_i$) ———*/
  for all $i \in S$ **do**
    update parent block maximum with v_i and F_i^{total}

  for all blocks b **do**
    compute new region membership per section 2

  for all $i \in S$ **do**
    decrement $validity_i$
    if ($validity_i \leq 0$) $\|$ (*parent block has a different time step*)
      set $compute_i =$ **true**
      update $validity_i$, $timestep_i$
    else
      $compute_i =$ **false**

  /*——— Physics computation ———*/
  for all $i \in S$ **do**
    if $compute_i$ **do** find neighborhoods N_i

  for all $i \in S$ **do**
    if $compute_i$ **do** update ρ_i, p_i

  for all $i \in S$ **do**
    if $compute_i$ **do** update $\mathbf{F}^{p,v,g}$

  for all $i \in S$ **do**
    predict new v_i
    predict new x_i
    if $compute_i$ **then**
      apply correction to v_i
      apply correction to x_i

IEEE Transactions on Visualization and Computer Graphics 99, PrePrints (2013), 1. 1

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), SCA '03, pp. 154–159. 1

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. rep. *Prog. Phys. 68* (2005), 1703–1759. 1

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Trans. Graph. 30*, 4 (July 2011), 81:1–81:8. 1

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Trans. Graph. 28*, 3 (July 2009), 40:1–40:6. 1

[SRS03] SERNA A., R. D.-T., SÃąIZ A.: Conservation laws in smooth particle hydrodynamics: The deva code. *The Astrophysical Journal 597*, 3 (July 2003), 597:878–597:892. 2, 3

[TH07] TAKAHIRO HARADA SEIICHI KOSHIZUKA Y. K.: Smoothed particle hydrodynamics on GPUs. In *Proc. of Computer Graphics International* (2007), pp. 63–70. 1

[ZSP08] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Adaptive sampling and rendering of fluids on the GPU. In *Proceedings of the Fifth Eurographics / IEEE VGTC conference on Point-Based Graphics* (2008), SPBG'08, pp. 137–146. 1